2022年ナレッジモール研究
最終報告

量子コンピューターの活用研究

―機械学習・量子化学計算・組み合わせ最適化への適用 –

2022-B-10-a

Agenda

1. メンバー一覧と役割

2. 研究活動の概要

3. 成果物紹介

   ①量子コンピューター初学者向け学習資料
   ②量子積分アルゴリズムの実装
   ③日本語手書き文字認識・筆跡鑑定回路の実装

4. まとめ

1. メンバー一覧と役割

氏名	会社名	役割	量子コンピューター学習経験年数（活動前）
茂木 一男	株式会社インテージテクノスフィア	メンバー	2年6ヶ月
加藤 誠司	キンドリルジャパンテクノロジーサービス株式会社	メンバー	1年6ヶ月
田中 凜太郎	大樹生命アイテクノロジー株式会社	リーダー	0年
小林 晃士	日本アイ・ビー・エム株式会社	メンバー	5ヶ月
永井 洋平	日本アイ・ビー・エム デジタルサービス株式会社	コーディネーター	3ヶ月
川島 貴司	株式会社フジミック	メンバー	2年6ヶ月
石塚 直満	リコーITソリューションズ株式会社	メンバー	5ヶ月
松島 輝昌	日本アイ・ビー・エム株式会社	サブリーダー	2ヶ月
沼田 祈史	日本アイ・ビー・エム株式会社	IBMアドバイザー	-
平山 毅	日本アイ・ビー・エム株式会社	IBMアドバイザー	-

2. 研究活動の概要

研究活動の進め方

当WGは**メンバーの半数以上が量子コンピューター学習経験1年未満の初学者**だということをふまえ、研究活動前半は**量子コンピューターに関する勉強会**を実施した。
勉強会の教材には、IBMがクラウド上で提供している量子コンピューター「**IBM Quantum**」を実行するためのインターフェースPythonプログラミングパッケージである「**Qiskit**」の学習用教材「**Qiskit textbook**」（https://qiskit.org/textbook/ja/preface.html）を主に使用した。
研究活動後半には量子コンピューターの学習を進める中で、メンバーそれぞれが取り組みたい課題を考えた結果、下記の3つのテーマに集約された。これらのテーマに沿った成果物の内容をメンバー全員で検討し、テーマ毎に**3つのチームに分かれて成果物の作成**を行った。

①量子人材の育成

②量子アルゴリズムの探究

③機械学習分野への応用

2. 研究活動の概要

研究会開催履歴

毎週火曜日　18:00 ～ 19:00で実施（5/3休会）

開催日	茂木	加藤	田中	小林	永井	川島	石塚	松島	沼田A	平山A
2月9日	○	○	○	○	○	○	○	○	○	×
2月15日	○	○	○	○	×	×	○	×	○	×
2月22日	○	○	○	○	○	○	○	○	○	○
3月1日	○	○	○	○	○	×	○	○	○	×
3月8日	○	×	○	○	×	○	○	○	○	○
3月15日	○	○	○	○	○	○	×	×	○	○
3月22日	○	○	○	○	×	×	×	○	○	×
3月29日	○	○	○	○	○	○	×	○	○	○
4月5日	○	○	○	○	○	○	○	×	○	×
4月12日	×	○	○	○	○	○	○	○	○	×
4月19日	○	○	○	○	○	○	○	○	○	×
4月26日	○	○	○	○	○	○	○	×	○	×
5月10日	○	○	○	○	○	○	○	○	○	×
5月17日	○	○	○	○	×	○	○	○	○	○
5月24日	○	○	○	○	×	×	×	○	○	×
5月31日	○	×	○	×	×	○	○	○	○	×
6月7日	○	○	○	○	×	○	○	○	○	×
6月14日	○	○	○	○	×	○	○	○	○	×
6月21日	○	○	○	○	×	×	○	○	○	×
6月28日	○	○	○	○	×	×	○	○	○	×
7月5日	○	○	○	○	×	○	○	○	○	×
7月12日	○	×	○	×	×	○	○	○	○	×

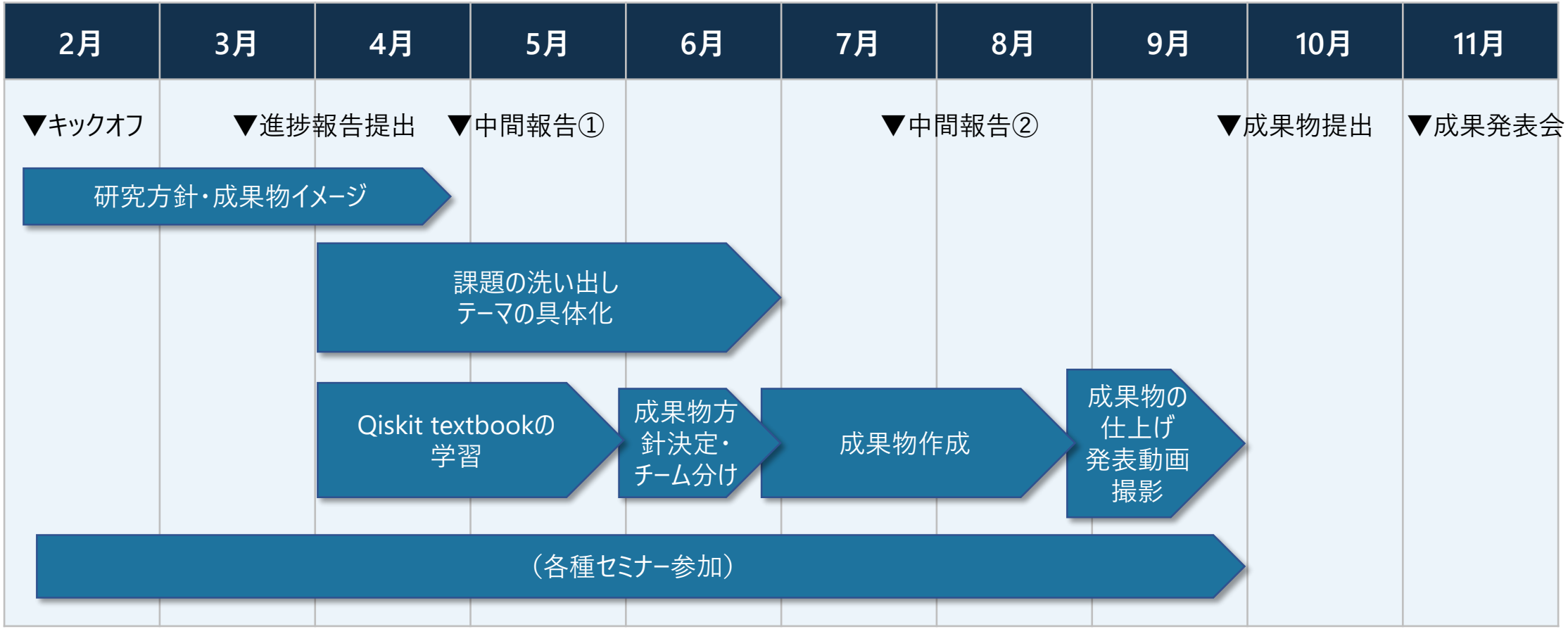
2. 研究活動の概要

研究会開催履歴

開催日	茂木	加藤	田中	小林	永井	川島	石塚	松島	沼田A	平山A
7月19日	○	○	○	○	×	○	×	○	○	×
7月26日	○	○	○	○	×	×	×	×	○	×
8月2日	○	○	○	○	×	○	○	○	×	×
8月9日	○	○	○	×	×	×	×	○	○	×
8月16日	○	○	○	○	×	○	○	×	○	×
8月23日	○	○	○	○	×	×	○	○	○	×
8月30日	○	○	×	○	×	×	×	○	○	×
9月6日	○	○	○	○	×	×	×	○	○	×
9月13日	○	○	○	×	×	×	○	○	○	×
9月20日	○	○	○	○	×	○	×	○	○	○
9月27日	○	○	○	○	×	○	○	○	○	×

2. 研究活動の概要

全体スケジュール



3. 成果物紹介

前述の3つのテーマに対し、各チームでそれぞれ次のような成果物を作成した。

①**量子コンピューター初学者向け学習資料** - 「量子人材の育成」チーム（加藤、田中、石塚、松島）
初学者が「Qiskit textbook」を使用して量子コンピューターを学習する際の補助となる学習教材、
および初学者が学習時に躓きやすい知識をまとめた知識集を作成した。

②**量子積分アルゴリズムの実装** - 「量子アルゴリズムの探究」チーム（茂木、沼田）
古典モンテカルロ積分の代替として、量子加速が得られる量子積分アルゴリズムについて、
現在単純に実装できることが知られている$\sin^2 x$系以外の関数の量子積分回路を実装した。

③**日本語手書き文字認識・筆跡鑑定回路の実装** - 「機械学習分野への応用」チーム（小林、川島）
量子ニューラルネットワークを用いた日本語手書き文字のMNIST分類回路、
および2者の筆跡判定回路の実装を行った。

①量子コンピューター初学者向け学習資料

初学者向け学習資料について

背景

今回の研究活動で初学者として量子コンピューターを学習して感じた点は以下の通り。

- 量子コンピューターの基礎的な原理について簡単に説明した入門資料や記事は数多く存在しているが、入門レベルから実際の量子アルゴリズムの理解に至るまでには多くの知識的ギャップが存在しており、その部分について同じレベルで簡単に説明された資料はあまり無い。
- 量子コンピューターの理解には、数学・物理学についての知識がある程度必要である。

目標

上記を踏まえ、以下2つの資料を作成することとした。

1) Qiskit textbookの学習補助教材：初学者が量子コンピューターを学習する際、前提となる数学知識から量子アルゴリズムまで体系的に学べ、かつWeb上で無料で公開されている「Qiskit textbook」を使用することを想定し、特に内容が難しいと思われる部分を分かりやすく解説する資料を作成した。
2) 初学者が学習時に躓く知識集：初学者が量子コンピューターを学習する際に躓くと思われる数学や物理学の知識集、および「Qiskit」で「IBM Quantum」を実行する際のポイントについてまとめた資料を作成した。

作成資料一覧

資料名	該当Qiskit textbookページ	担当
量子とは？（マイナスの確率）	量子とは？（qiskit.org）	松島
1.3量子ビット状態を表現する（ブロッホ球）	量子ビット状態を表現する（qiskit.org）	加藤
2.3位相キックバック	位相キックバック（qiskit.org）	田中
3.5量子フーリエ変換（DFTの基礎）	量子フーリエ変換（qiskit.org）	田中
3.5量子フーリエ変換	量子フーリエ変換（qiskit.org）	加藤
3.6量子位相推定	量子位相推定（qiskit.org）	田中
3.8グローバーのアルゴリズム	グローバーのアルゴリズム（qiskit.org）	加藤
3.9量子数え上げ	量子数え上げ（qiskit.org）	田中
初学者が学習時に躓く知識集	-	石塚

資料作成箇所選定のポイント

- 初学者にとって理解が難しい部分であること
- 当WGで作成する量子積分アルゴリズム（量子数え上げの応用）の理解に必要な知識をカバーすること

作成資料紹介

2.3 位相キックバック

位相キックバックとは

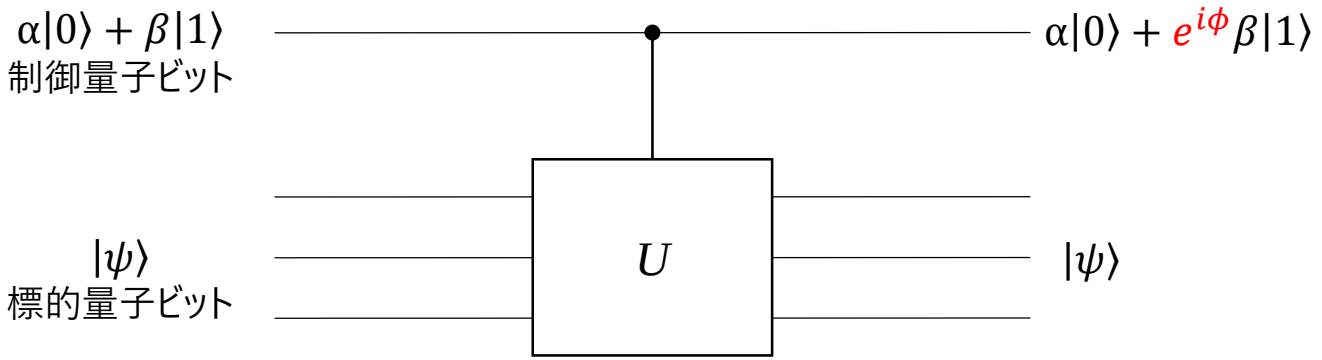
Qiskit textbookでは位相キックバックの簡単な例から紹介することで説明していますが、
ここでは位相キックバックがどういうものかをまず頭に入れるため、一般的な形から紹介します。

前提

あるユニタリー演算子Uに対して、固有状態$|\psi\rangle$と固有値$e^{i\phi}$が存在している。

$$U|\psi\rangle = e^{i\phi}|\psi\rangle$$

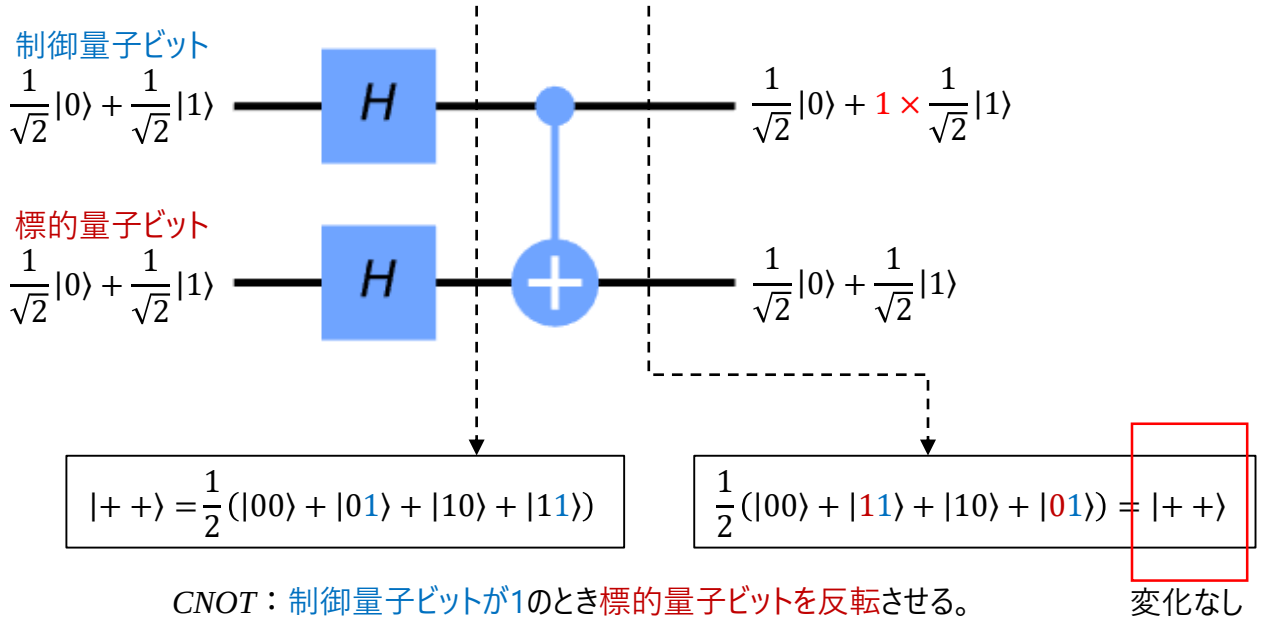
この固有状態$|\psi\rangle$を標的量子ビット、重ね合わせた量子ビット$\alpha|0\rangle + \beta|1\rangle$を制御量子ビットとして
制御ユニタリーゲートを作用させると、固有値$e^{i\phi}$が制御量子ビットの相対位相として現れる。
このとき標的量子ビットの状態は変化しない。



この前提でtextbookの例を見ていきます。

例1

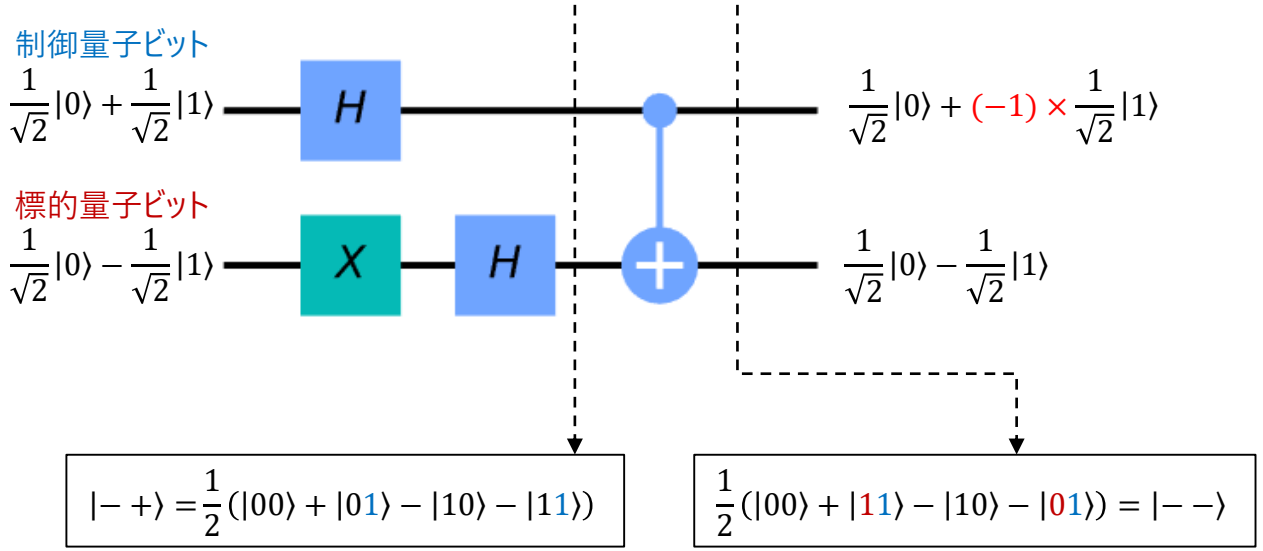
ユニタリー演算子$U = X$（∴制御ユニタリーゲートは$CNOT$）
固有状態$|\psi\rangle = |+\rangle$
固有値は$X|+\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = |+\rangle$ となるので1
制御量子ビット$\alpha|0\rangle + \beta|1\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle = |+\rangle$



固有値1の例だと何も変わりません。

例2

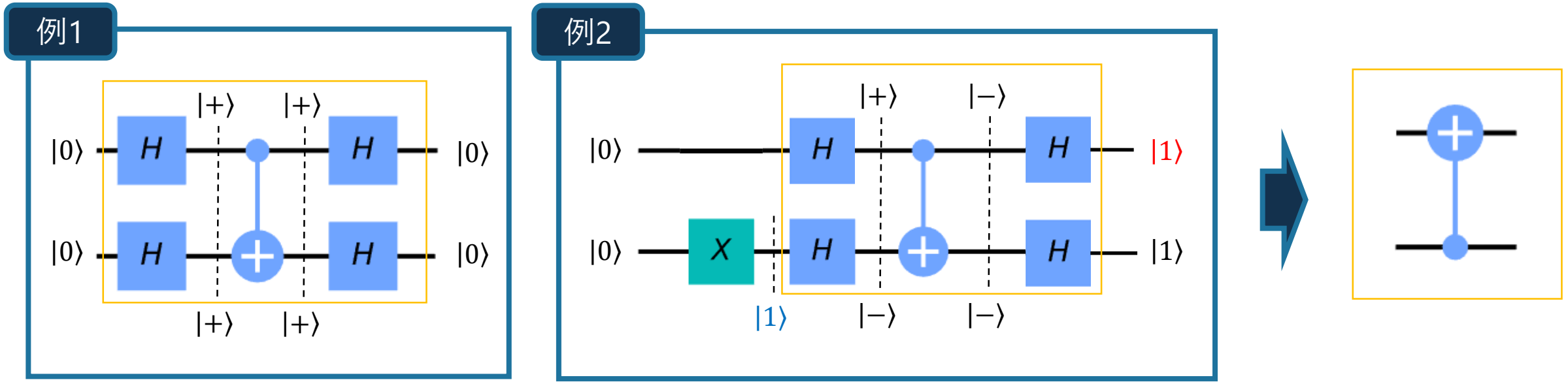
ユニタリー演算子$U=X$（∴制御ユニタリーゲートは$CNOT$）
固有状態$|\psi\rangle = |-\rangle$
固有値は$X|-\rangle = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}1\\ -1\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}-1\\ 1\end{pmatrix} = -|-\rangle$ となるので－1
制御量子ビット$\alpha|0\rangle + \beta|1\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle = |+\rangle$



固有値-1の例では確かに標的量子ビットを変化させずに、
固有値が制御量子ビットの位相に反映（キックバック）されています。

ゲートの等価性

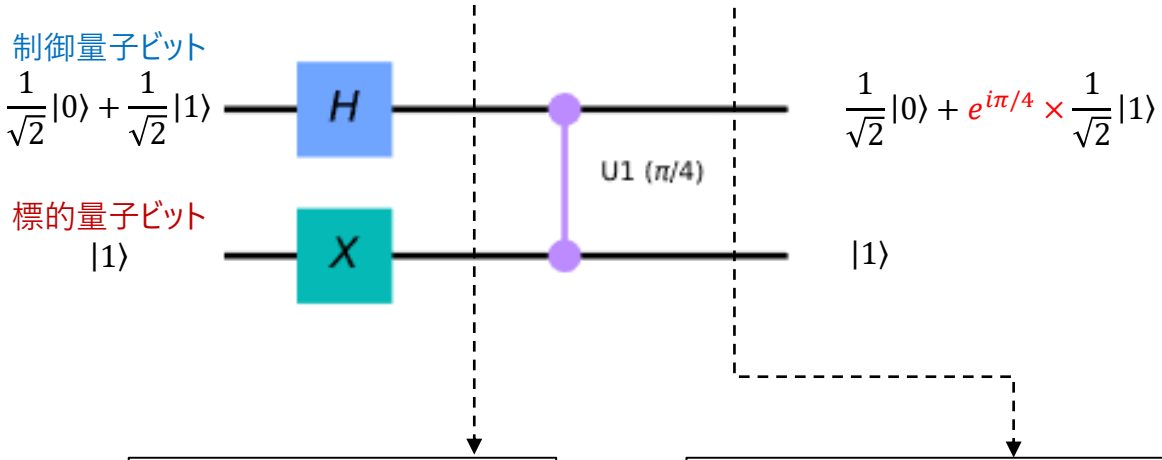
例1,2の状態で、更にアダマールゲートを適用すると、最終的な結果はどうなるでしょうか。



橙色で囲われた部分のゲート操作の結果は、CNOTゲートの制御量子ビットと標的量子ビットを入れ替えた結果になっていることが分かります。Qiskitのプログラム上では簡単に制御量子ビットと標的量子ビットを入れ替えられますが、一部の量子コンピュータではハードウェア的に逆向きのCNOT操作は実行できないため、位相キックバックを応用して逆向きのCNOT操作を実現しています。このような複数のゲートの組み合わせで他のゲートの動きを再現する、量子ゲートの等価性を上手く利用して様々な量子ゲートが実装されています。

例3

ユニタリー演算子$U = T$（∴制御ユニタリーゲートは*Controlled-T*）
固有状態$|\psi\rangle = |1\rangle$
固有値は$T|1\rangle = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = e^{i\pi/4}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = e^{i\pi/4}|1\rangle$ となるので$e^{i\pi/4}$
制御量子ビット$\alpha|0\rangle + \beta|1\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle = |+\rangle$



$$|1+\rangle = |1\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$
$$= \frac{1}{\sqrt{2}}(|10\rangle + |11\rangle)$$
$$= \frac{1}{\sqrt{2}}\begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\pi/4} \end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 0 \\ 0 \\ 1 \\ e^{i\pi/4} \end{pmatrix}$$
$$= \frac{1}{\sqrt{2}}(|10\rangle + e^{i\pi/4}|11\rangle)$$
$$= |1\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi/4}|1\rangle)$$

②量子積分アルゴリズムの実装

活動テーマ 量子積分

メンバー：茂木、沼田

取り上げる課題

- 量子加速が得られるような積分を行うアルゴリズムにおいて、被積分関数の選択肢の拡大可能性を探究します。
- 量子積分アルゴリズムは古典モンテカルロ積分の代替であり様々な分野での応用が考えられますが、現在は特に金融工学において金融商品の価格予測問題への適用が期待され、盛んに研究されています。
- 但し現状では積分できる関数に対して数学的に強い制約が存在し、これを緩和しようとすると必要なゲート数が急速に増大してしまいます。（q量子ビットに対して$O(2^q)$のCNOTゲート）
- この要因の一つは、量子ビットを被積分関数の関数値を表現するように振幅エンコーディングする部分です。

目標

完全に任意の関数の積分回路を構成することは現状では難しいと考えられるので、現状でも例外的に単純に回路が構成できる$sin^2 x$系の関数よりはある程度一般化された（或いは修飾が加えられた）、関数の積分回路構成を目指します。

1　量子数値積分 概要

$f(x)$は既知である(ゲートで構成できる)場合に、未知の$\int f(x)dx$を求めることが課題です。

- $\int f(x)dx$ に含まれる解の状態$|\psi_1\rangle$と、そうではない状態$|\psi_0\rangle$ (これらは正規直交)を考え、

$$|\psi\rangle = \cos\theta|\psi_0\rangle + e^{i\varphi}\sin\theta|\psi_1\rangle$$

  とします。

- 上記の状態があるときに偏角を2θ回す回転ゲートQを構成する方法が知られています(Quantum Amplitude Amplification で用いられる)
- このときQの固有値の位相は2θに等しくなります。
- よってQが構成出来れば、これにQPEを適用して固有値の位相を読みだすことによりθが求まります。

参考文献：　（量子コンピュータを用いた高速数値積分　宇野隼平）
https://qiita.com/KeiichiroHiga/items/dae9b5395366f0f4fc2c

1.1　被積分関数の定義

- 被積分関数を $S(f) = \sum_{x=0}^{2^n-1} p(x)\, f(x)$ とします。
- 二つの演算子　$\mathcal{P}, \mathcal{R}$ を定義します。

$$\mathcal{P}|0\rangle_n \coloneqq \sum_{x=0}^{2^n-1} \sqrt{p(x)}|x\rangle_n$$

$$\mathcal{R}|x\rangle_n|0\rangle \coloneqq |x\rangle_n \left(\sqrt{f(x)}|0\rangle + \sqrt{1-f(x)}|1\rangle\right)$$

- これを組み合わせた $\mathcal{A} \equiv \mathcal{R}(\mathcal{P} \otimes \mathcal{I})$ を初期状態 $|0>_{n+1}$ に作用させると

$$\mathcal{A}|0\rangle_{n+1} = \sum_{x=0}^{2^n-1} \sqrt{p(x)}|x\rangle_n \left(\sqrt{f(x)}|0\rangle + \sqrt{1-f(x)}|1\rangle\right)$$

- これを $S(f)$ で書き直すと

$$\mathcal{A}|0\rangle_{n+1} = \sqrt{S(f)}|\Psi_0\rangle_{n+1} + \sqrt{1-S(f)}|\Psi_1\rangle_{n+1}$$

と、なります。ここで

$$|\Psi_0\rangle_{n+1} \coloneqq \sum_{x=0}^{2^n-1} \sqrt{p(x)}\sqrt{\frac{f(x)}{S(f)}}|x\rangle_n|0\rangle$$

$$|\Psi_1\rangle_{n+1} \coloneqq \sum_{x=0}^{2^n-1} \sqrt{p(x)}\sqrt{\frac{1-f(x)}{1-S(f)}}|x\rangle_n|1\rangle$$

です。

1.2　振幅増幅演算子

- 振幅増幅演算子は $\mathcal{Q} \coloneqq \mathcal{U}_{\Psi}\mathcal{U}_{\Psi_0}$　です。ここで

$$\mathcal{U}_{\Psi_0} \coloneqq I_n \otimes Z$$

$$\mathcal{U}_{\Psi} \coloneqq \mathcal{A}\left(I_{n+1} - 2|0\rangle_{n+1}\langle 0|_{n+1}\right)\mathcal{A}^{\dagger}$$

- この$\mathcal{Q}$を$\mathcal{A}|0\rangle_{n+1}$に作用させると

$$\mathcal{Q}^j\mathcal{A}|0\rangle_{n+1} = \cos\{(2j+1)\theta\}|\Psi_0\rangle_{n+1} + \sin\{(2j+1)\theta\}|\Psi_1\rangle_{n+1}$$

  となります。

- このとき $\mathcal{Q}$の固有値と固有ベクトルは

$$\lambda = e^{\pm 2i\theta}$$

$$|\Psi_{\pm}\rangle_{n+1} = \frac{1}{\sqrt{2}}(\Psi_0\rangle_{n+1} \mp i|\Psi_1\rangle_{n+1})$$

$$\mathcal{A}|0\rangle_{n+1} = \frac{1}{\sqrt{2}}\left(e^{i\theta}\left|\Psi_+\right\rangle_{n+1} + e^{-i\theta}\left|\Psi_-\right\rangle_{n+1}\right)$$

  となります。

1.3　振幅推定 (1)

- $|+\rangle_m = \frac{1}{\sqrt{M}}\sum_{x=0}^{M-1}|x\rangle_m$ として

$$|+\rangle_m \mathcal{A}|0\rangle_{n+1} = \frac{1}{\sqrt{M}}\sum_{x=0}^{M-1}|x\rangle_m \frac{1}{\sqrt{2}}\left(e^{i\theta}|\Psi_+\rangle_{n+1} + e^{-i\theta}|\Psi_-\rangle_{n+1}\right)$$

- ここで $M = 2^m$ です。各$|x\rangle_m$から$\mathcal{A}|0\rangle_{n+1}$に制御ユニタリ行列$\mathcal{CQ}$を実施すると、

$$\mathcal{CQ}(|+\rangle_m \mathcal{A}|0\rangle_{n+1}) = \frac{e^{i\theta}}{\sqrt{2M}}\sum_{x=0}^{M-1} e^{2ix\theta}\,|x\rangle_m|\Psi_+\rangle_{n+1} + \frac{e^{i\theta}}{\sqrt{2M}}\sum_{x=0}^{M-1} e^{-2ix\theta}|x\rangle_m|\Psi_-\rangle_{n+1}$$

となります。

- ここで $|S_M(x)\rangle$を次のように定義します

$$|S_M(x)\rangle \coloneqq \frac{1}{\sqrt{M}}\sum_{y=0}^{M-1} e^{2\pi ixy}|y\rangle$$

$$|S_M(x)\rangle = |S_M(k+x)\rangle, (k \in \mathbb{Z})$$

1.3　振幅推定 (2)

- x が整数の場合には

$$F_m^{-1}|\,S_M\left(\frac{x}{M}\right)\rangle = |x\rangle$$

  です。

- 逆フーリエ変換を行うと

$$\frac{e^{i\theta}}{\sqrt{2}}F_m^{-1}|\,S_M\left(\frac{\theta}{\pi}\right)\rangle_m|\Psi_+\rangle_{n+1} + \frac{e^{-i\theta}}{\sqrt{2}}F_m^{-1}|\,S_M\left(1-\frac{\theta}{\pi}\right)\rangle_m|\Psi_-\rangle_{n+1}$$

- ここでmビットを観測すると、$|0\rangle_m \cdots |M-1\rangle_m$のうち

$$F_m^{-1}\,|S_M\left(\frac{\theta}{\pi}\right)\rangle_m \approx |\frac{M}{\pi}\theta\rangle_m, F_m^{-1}\,|S_M\left(1-\frac{\theta}{\pi}\right)\rangle_m \approx |\,M\left(1-\frac{\theta}{\pi}\right)\rangle_m$$

となります。

2.1　$\sin^2 x$ の場合の量子数値積分

- $sin^2 x$ の場合は

$$S(f) = \sum_{x=0}^{2^n-1} \frac{1}{2^n} sin^2 \frac{x}{2^n}, \sum_{x=0}^{2^n-1} \frac{1}{2^n} = 1$$

であり、

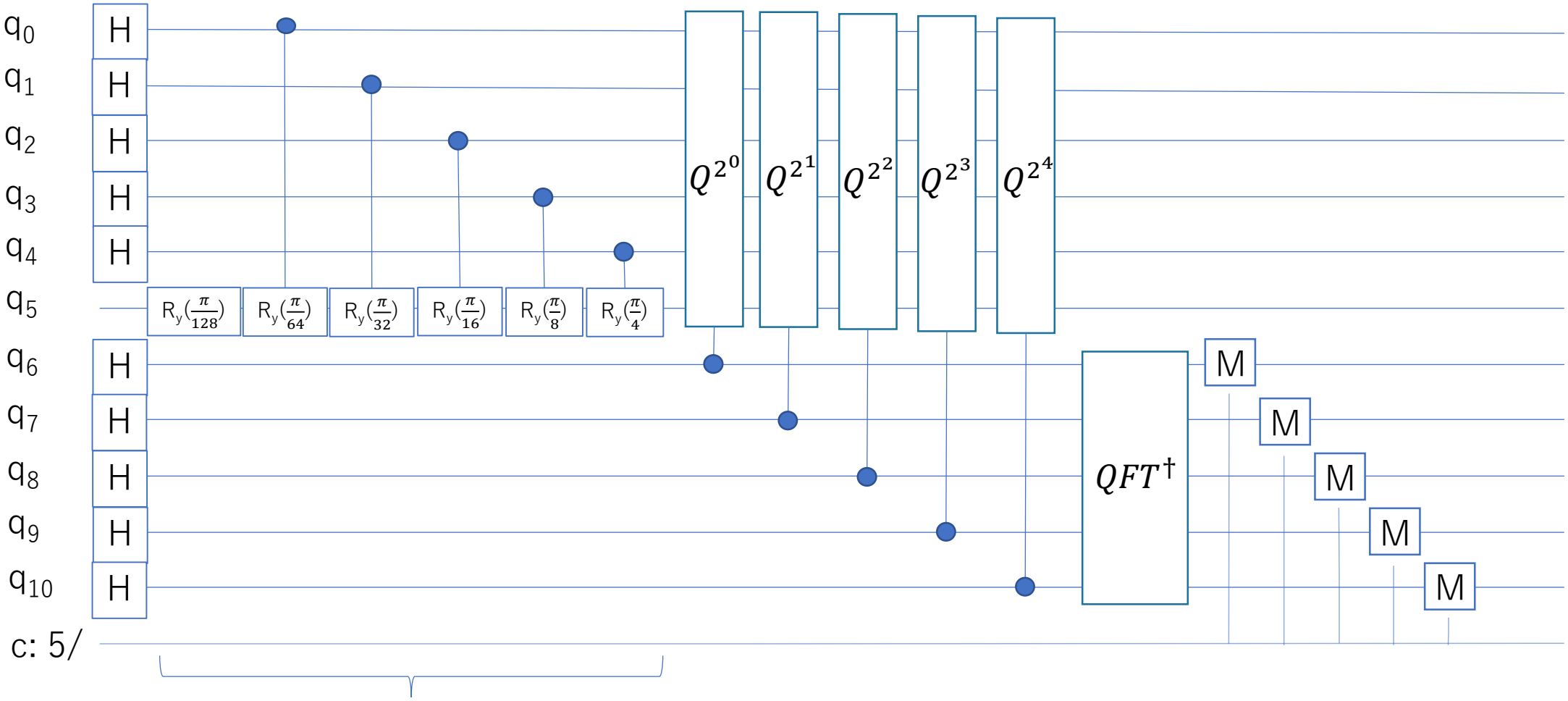
- このとき $\mathcal{A} \coloneqq \mathcal{R}(\mathcal{P} \otimes \mathcal{I})$ は

$$\mathcal{P}|0\rangle_n \coloneqq \sum_{x=0}^{2^n} \sqrt{\frac{1}{2^n}} |x\rangle_n = H^{\otimes n}|x\rangle_n$$

$$\mathcal{R}|x\rangle_n|0\rangle \coloneqq |x\rangle_n \left(\sin \frac{x}{2^n} |0\rangle + \cos \frac{x}{2^n} |1\rangle \right)$$

となって非常に簡単になります。

2.2　回路図 ($\int \sin^2 x \, dx$)

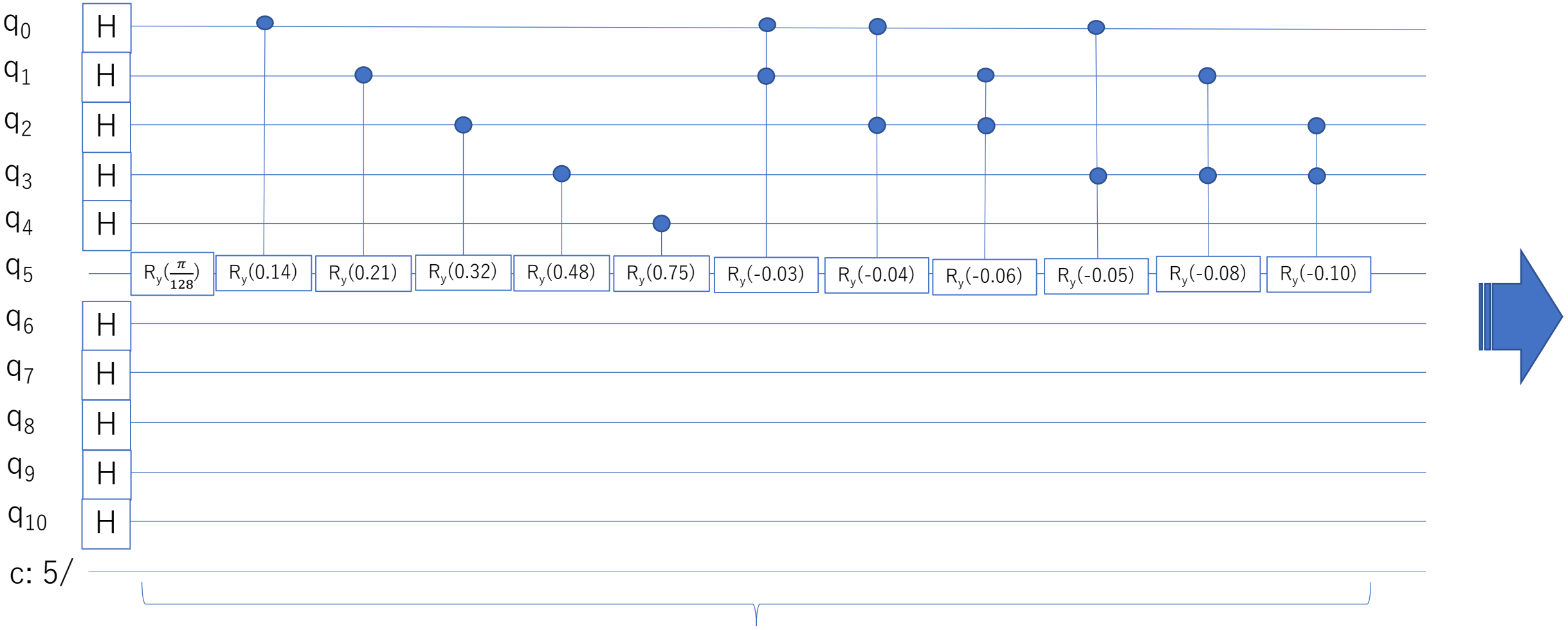


被積分関数を設定 $\mathcal{R}$

3.1　一般の場合の量子数値積分

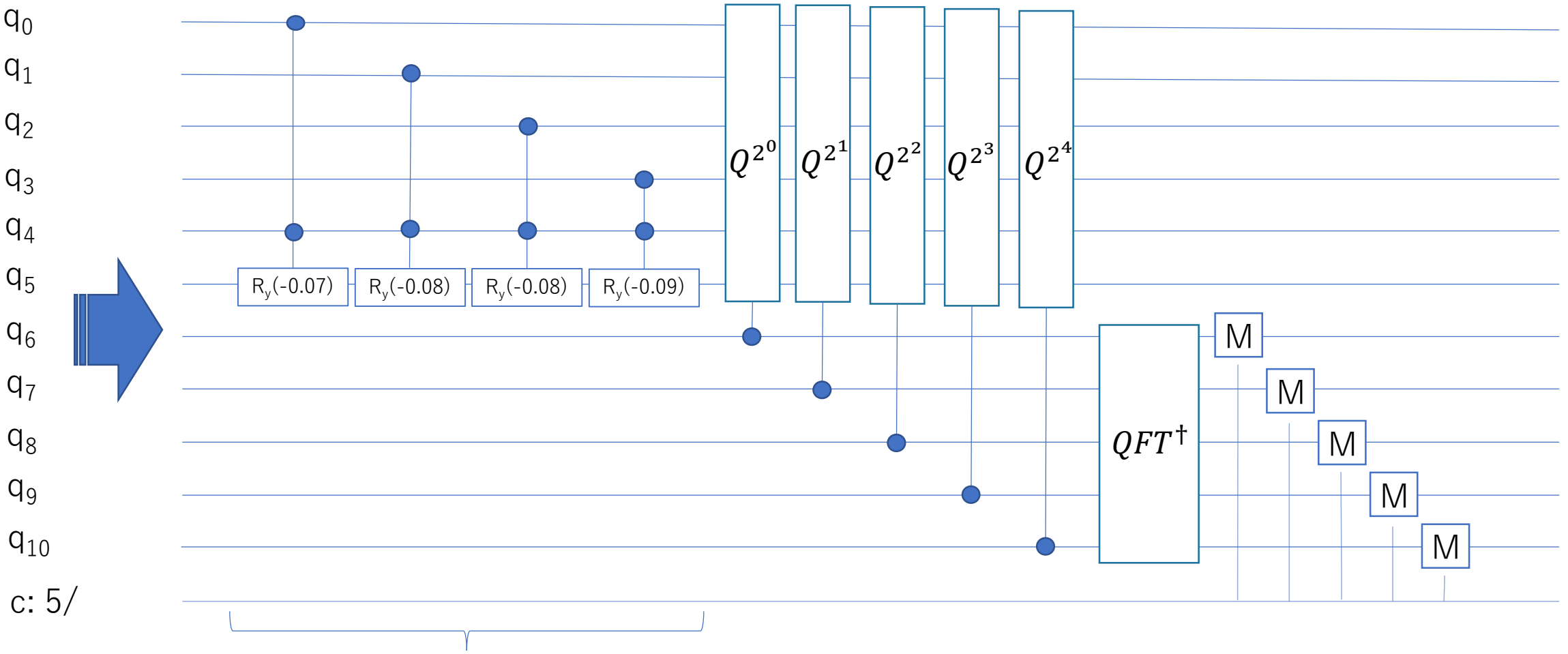
- $\mathcal{P}$や$\mathcal{R}$は量子コンピュータでは回転ゲートを用いて構成されるので、一般の $f(x)$ の積分の場合にはこれを正確に設定するのに多数のゲートが必要になります。
- 今回は最大5qubitの精度で積分を行います。これで実現できる精度は十進換算で1桁程度ですが、これ以下の精度だと結果検証の信頼性が低くなるからです。
- そこで、この精度範囲で大きな支障のない範囲内で $f(x)$ を設定することにし、ccr_y で構成できる範囲で x などの関数を表現することを考えました。
- また、各積分区間の関数値に相当する回転角は、古典的に予め計算して計算して設定しました。
- 計算される積分値には多少の誤差が生じますが、今回計算に使った 5qubit で表現できる精度の範囲内では、十分許容される範囲となります。
- 積分区間は、0 ～ π／2 としました。

3.2　回路図 (∫ x dx)　(1)



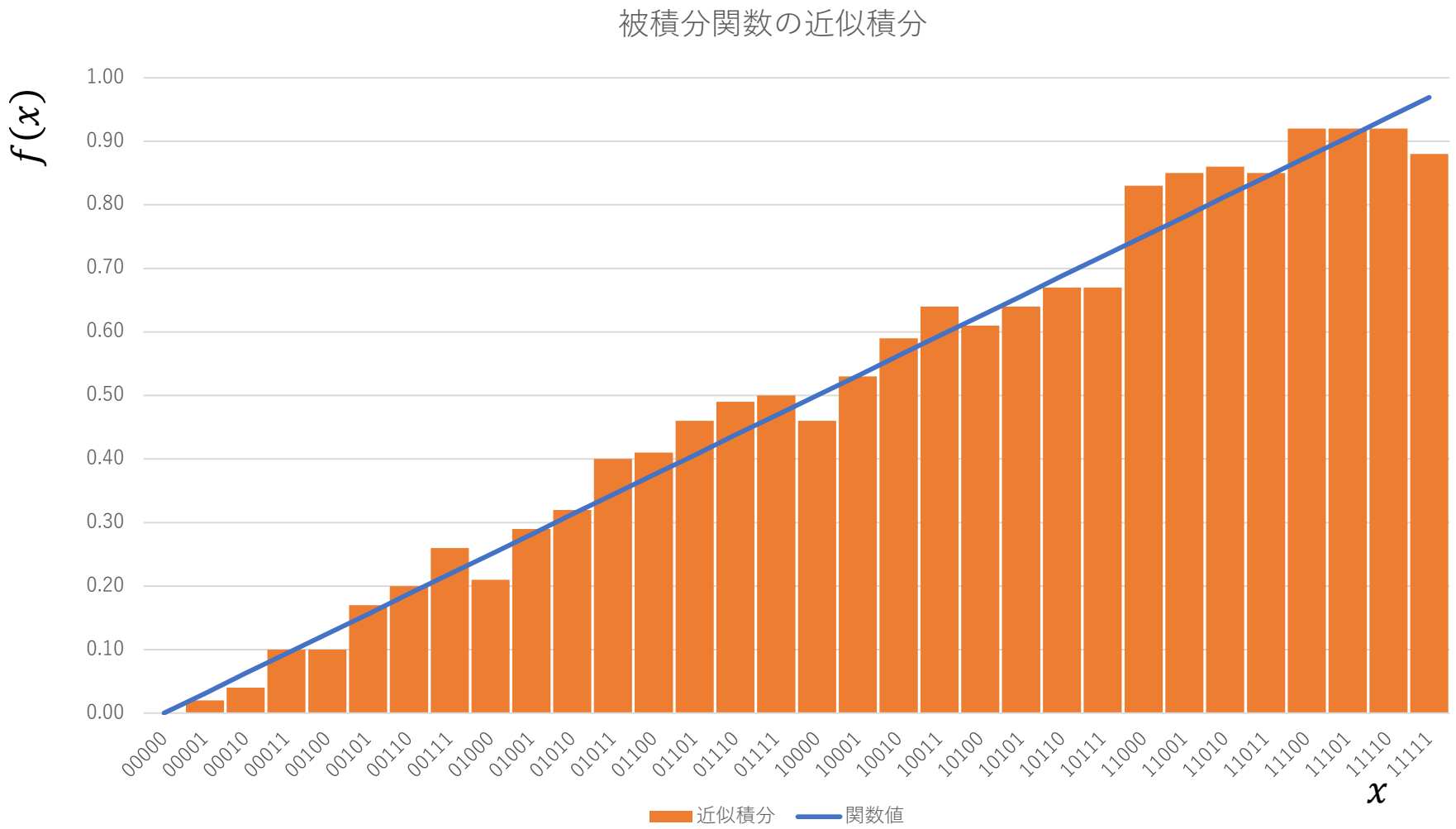
被積分関数を設定 $\mathcal{R}$

3.2　回路図 (∫ x dx)　(2)

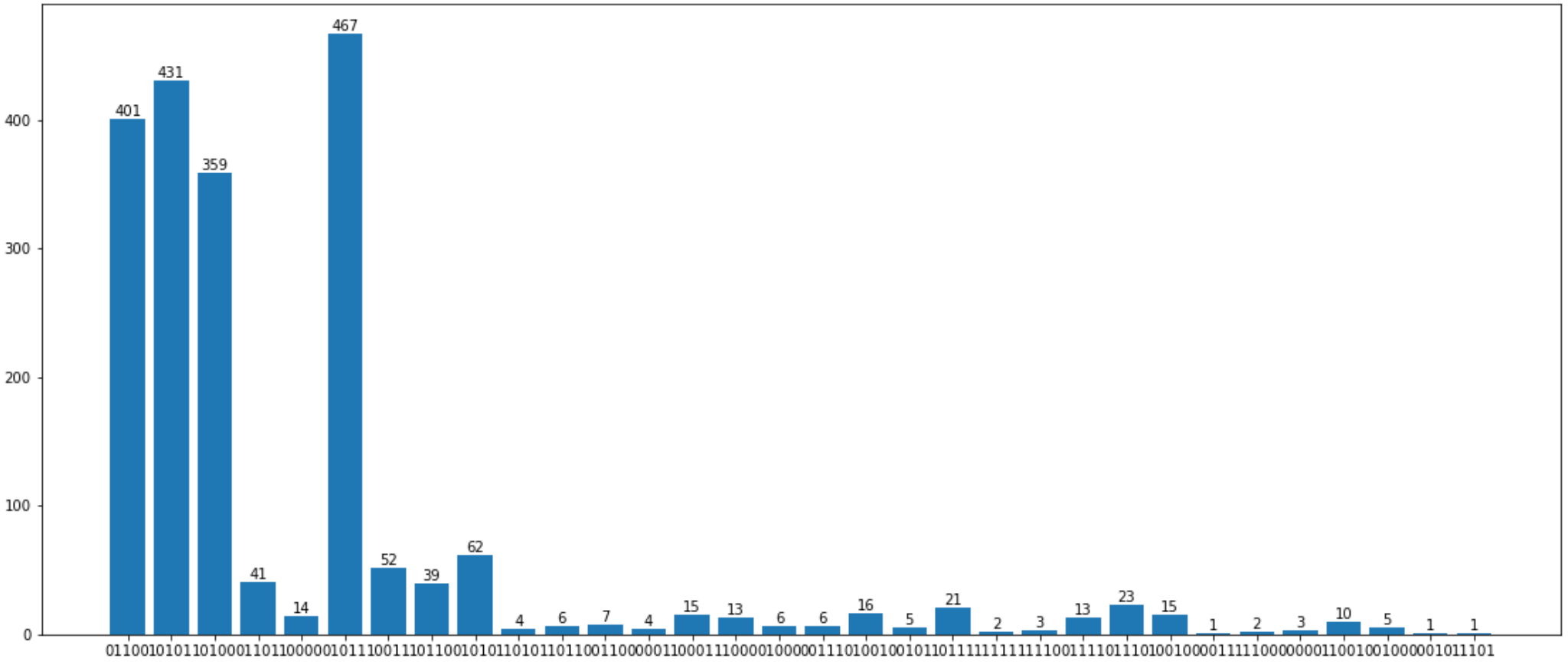


被積分関数を設定 $\mathcal{R}$

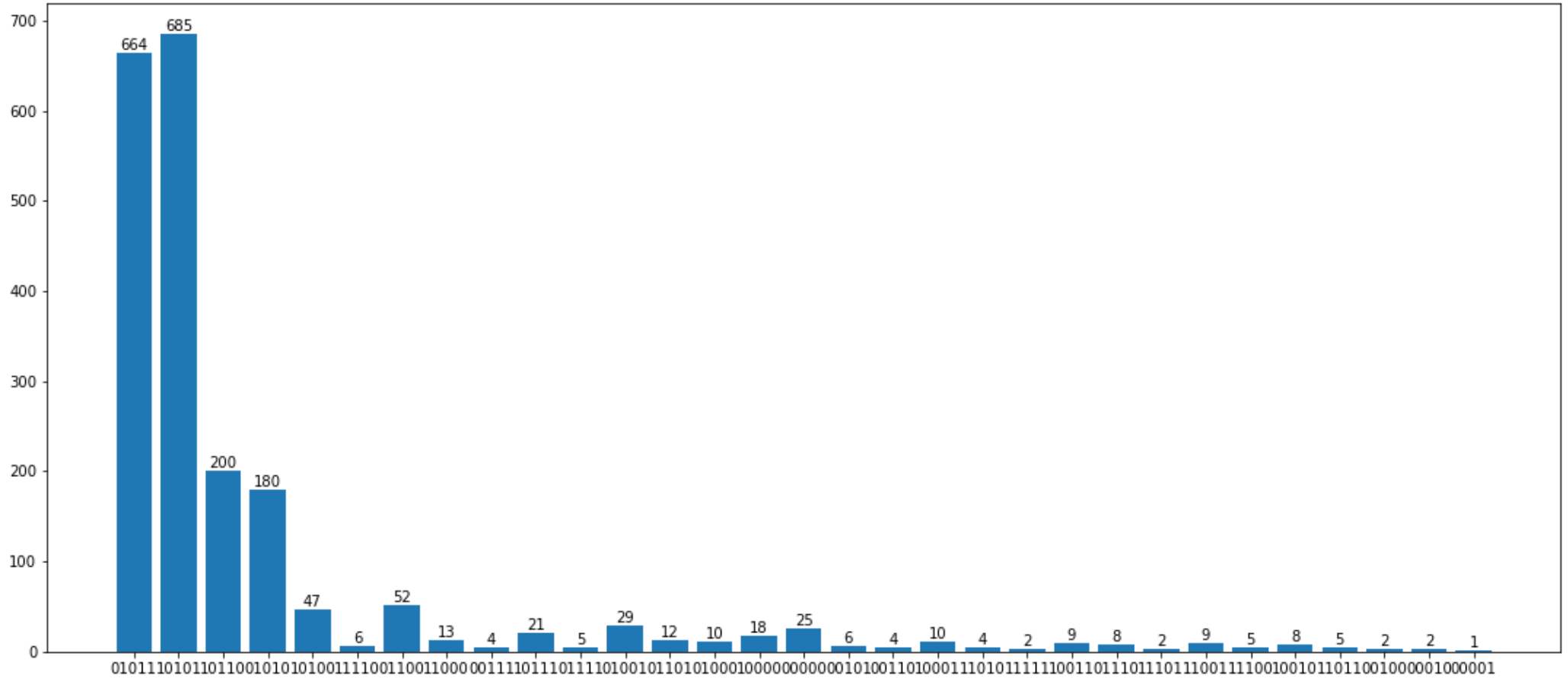
3.3 ccr_y による被積分関数の近似



4.1　計算結果 ($\int \sin^2 x \, dx$)



4.2　計算結果 (∫ x dx)


700
600
500
400
300
200
100
0
664
685
200
180
47
6
52
13
4
21
5
29
12
10
18
25
6
4
10
4
2
9
8
2
9
5
8
5
2
2
1
0101110101101100101010100111100110011000001111011101111010001011010100010000000000101001101000111010111111001101110111011011100111100100101101100100000100000 1

4.3　解析解と計算値

積分した関数	原始関数		積分値	計算値
$sin^2(x)$	$\frac{x}{2} - \frac{\sin 2x}{4} + C$	積分区間 $0 \sim \frac{\pi}{2}$	0.7854	0.70 (5qubit)
$0\ \left(x < \frac{\pi}{2}\right)$	$C\ \left(x < \frac{\pi}{2}\right)$			
$1\ \left(x \geq \frac{\pi}{2}\right)$	$x + C\ \left(x \geq \frac{\pi}{2}\right)$			
$\frac{2}{\pi}x$	$\frac{1}{\pi}x^2 + C$		0.7854	0.70 (5qubit)
$\left(\frac{2}{\pi}\right)^2 x^2$	$\frac{4}{3\pi^2}x^3 + C$		0.5236	0.46 (4qubit)

③日本語手書き文字認識・筆跡鑑定回路の実装

量子機械学習 –ハイブリッドQNNで筆跡鑑定にトライ！-

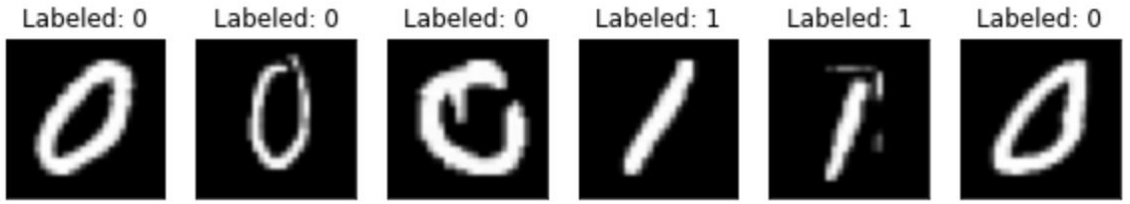
アジェンダ

1. MNIST分類のデータ形式[参考]
2. ハイブリッドQNNのQiskit実装[基礎編]
3. ハイブリッドQNNで筆跡鑑定[応用編]

1-1. MNIST分類 -データローダー-

MNISTデータセット

- ラベル：0,1
- 画像サイズ：28x28ピクセル



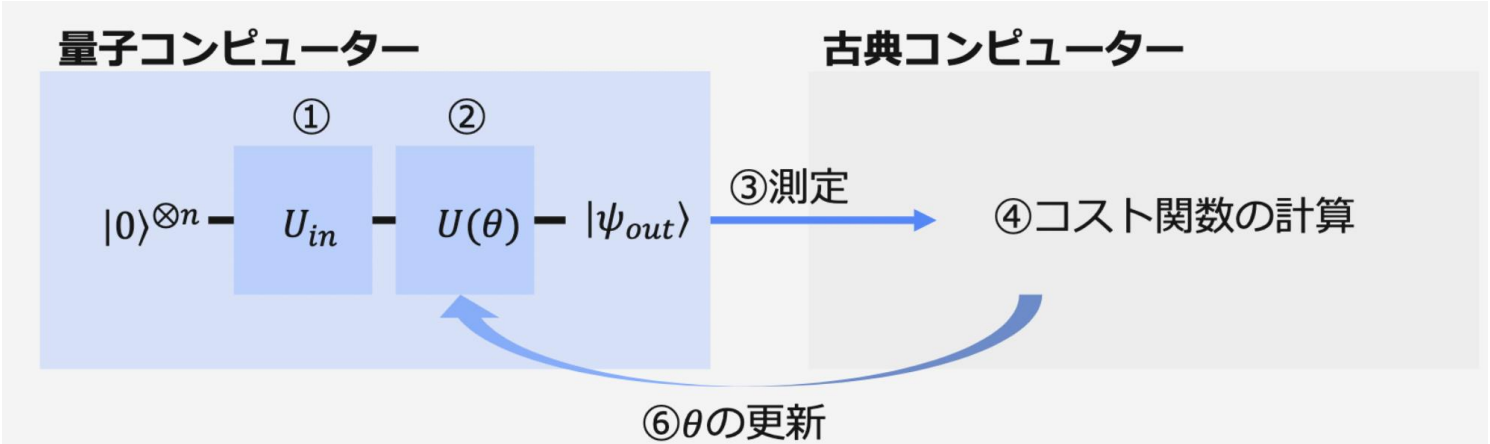
MNISTデータローダー

- Pytorchではデータセットをデータローダーに変換して学習
- 28x28の行列で値は0～1にしてイテレータとして呼び出す

2-1. ハイブリッドQNN -QNNについて-

量子ニューラルネットワーク(QNN; Quantum Neutral Network）について

- 量子ニューラルネットワークは、量子回路を学習モデルに見立てて機械学習タスクを行う。その際に、重ね合わせや量子もつれといった量子特有の特性を活用して、古典コンピュータでは表現が困難な非線形モデルを効率的に構築されることが期待される。

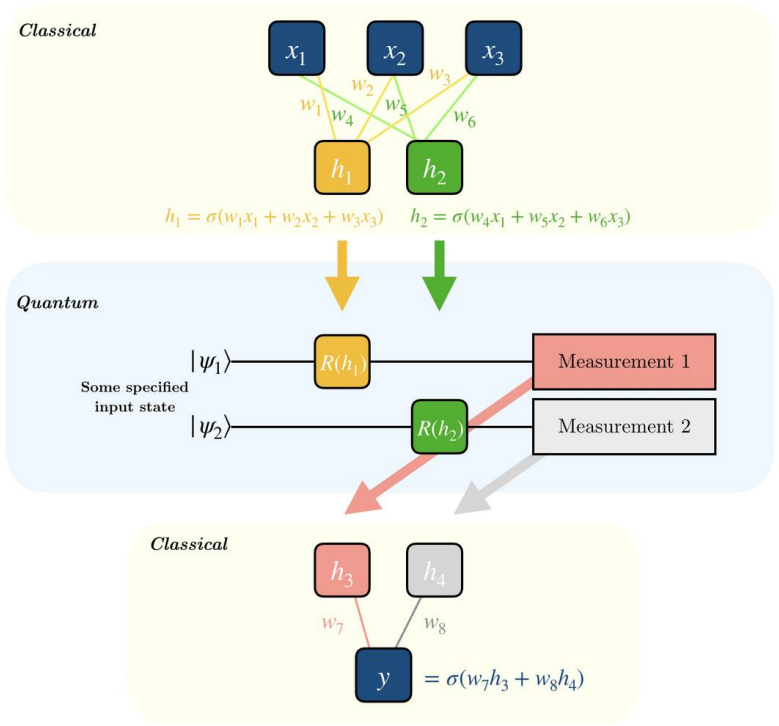


①入力値$\boldsymbol{x}$をパラメータに持つ量子回路U_{in}を通して量子状態にエンコード
②パラメータθに持つ変分回路を通して出力状態を生成
③量子ビットを測定して出力を求める
④コスト関数を計算する
⑤コスト関数が小さくなるようにθを更新して最適化θ^*を求める

2-2. ハイブリッドQNN -量子回路-

量子古典ニューラル・ネットワークによる実装について

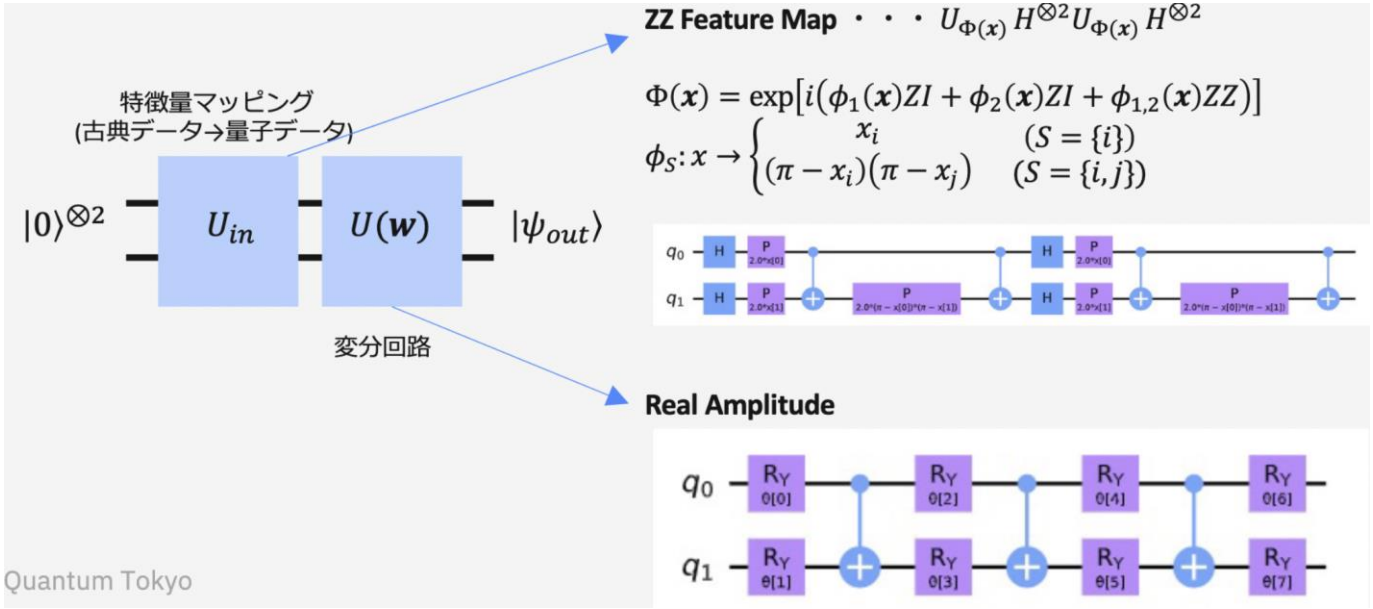
- 量子古典ニューラル・ネットワークを構築するには、パラメータ化された量子回路を使用してニューラル・ネットワークの隠れ層を実装する方法がある。

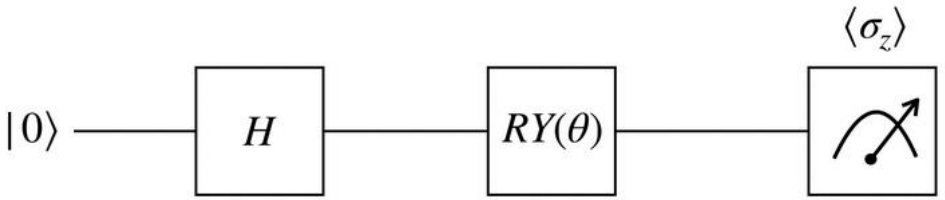


2-3. ハイブリッドQNN -量子分類-

Qiskitを用いた「量子クラス」の作成について

- 量子パラメータの数と、量子回路で使用したいショットの数を指定することで、量子関数をクラスに導入することができる





2-4. ハイブリッドQNN -量子古典クラス-

Qiskitを用いた「量子古典クラス」の作成について

- 順伝播のための量子回路を定義できたので、以下のように誤差逆伝播法に必要な関数を作成する。

```
# Define and create QNN
def create_qnn():
    feature_map = ZZFeatureMap(2)
    ansatz = RealAmplitudes(2, reps=1)
    # REMEMBER TO SET input_gradients=True FOR ENABLING HYBRID GRADIENT BACKPROP
    qnn = TwoLayerQNN(
        2,
        feature_map,
        ansatz,
        input_gradients=True,
        exp_val=AerPauliExpectation(),
        quantum_instance=qi,
    )
    return qnn


qnn4 = create_qnn()
print(qnn4.operator)
```

```
ComposedOp([
  OperatorMeasurement(1.0 * ZZ),
  CircuitStateFn(
  q_0: ┤0                          ├┤0                              ├
       │  ZZFeatureMap(x[0],x[1]) ││  RealAmplitudes(θ[0],θ[1],θ[2],θ[3]) │
  q_1: ┤1                          ├┤1                              ├
  )
])
```

2-5. ハイブリッドQNN -構築-

ハイブリッド・ニューラル・ネットワークの作成

- 赤枠の箇所が置き換わっている

```
# Define torch NN module


class Net(Module):
    def __init__(self, qnn):
        super().__init__()
        self.conv1 = Conv2d(1, 2, kernel_size=5)
        self.conv2 = Conv2d(2, 16, kernel_size=5)
        self.dropout = Dropout2d()
        self.fc1 = Linear(256, 64)
        self.fc2 = Linear(64, 2)  # 2-dimensional input to QNN
        self.qnn = TorchConnector(qnn)  # Apply torch connector, weights chosen
        # uniformly at random from interval [-1,1].
        self.fc3 = Linear(1, 1)  # 1-dimensional output from QNN

    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = F.max_pool2d(x, 2)
        x = F.relu(self.conv2(x))
        x = F.max_pool2d(x, 2)
        x = self.dropout(x)
        x = x.view(x.shape[0], -1)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        x = self.qnn(x)  # apply QNN
        x = self.fc3(x)
        return cat((x, 1 - x), -1)


model4 = Net(qnn4)
```

2-6. ハイブリッドQNN -学習-

最適化

- 最適化（Optimization）とは、ニューラルネットワークにおいて損失を極限まで最小化できていることを指す。Qiskitでは、Adam最適化ツールなどを使用することができる。

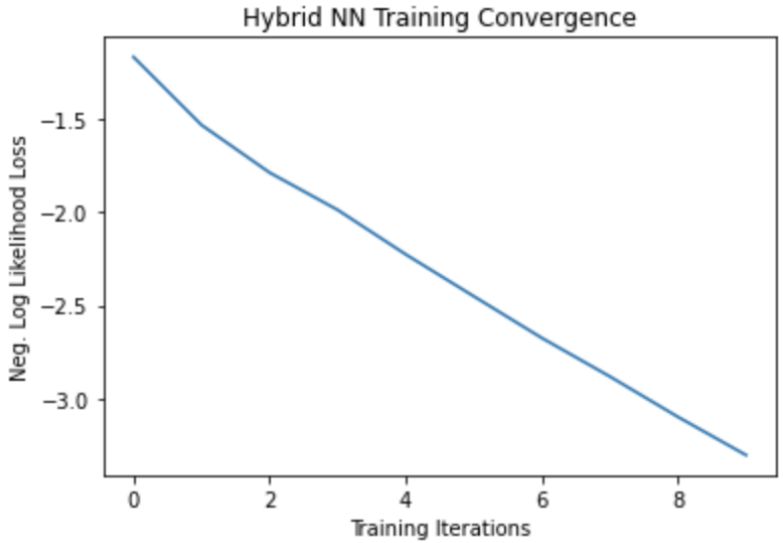
学習率

- 学習率とは、損失関数が最小となるパラメータをどの程度移動させるかの割合を示す。

損失関数

- 平均二乗誤差などの計算結果の数値のことを損失（Loss）と呼び、この損失を求めるための関数のことを損失関数（Loss Function）と呼ぶ。Qiskit上では負の対数尤度損失関数などを使用している。

```
Training [10%]  Loss: -1.1630
Training [20%]  Loss: -1.5294
Training [30%]  Loss: -1.7855
Training [40%]  Loss: -1.9863
Training [50%]  Loss: -2.2257
Training [60%]  Loss: -2.4513
Training [70%]  Loss: -2.6758
Training [80%]  Loss: -2.8832
Training [90%]  Loss: -3.1006
Training [100%] Loss: -3.3061
```

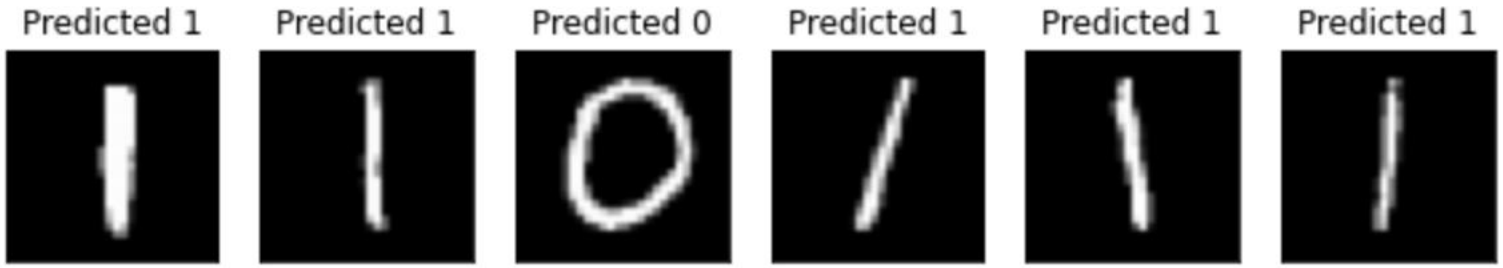


2-7. ハイブリッドQNN -評価-

ネットワークのテスト・評価

- スコア(損失と正確性)と検証用データの推論結果は以下。

```
Performance on test data:
Loss: -3.3585
Accuracy: 100.0%
```



高い精度を出すことができている

量子機械学習（QNN）で一般問題が解ける？

3. 筆跡鑑定

アジェンダ

1) 筆跡鑑定について
2) 学習データ
3) 学習
4) 評価

3-1. 筆跡鑑定 -筆跡鑑定について-

筆跡鑑定とは

- 複数の筆跡を比較し、それを書いた筆者が同一人であるか別人であるかを識別します。

なぜ、筆跡鑑定にトライしたか？

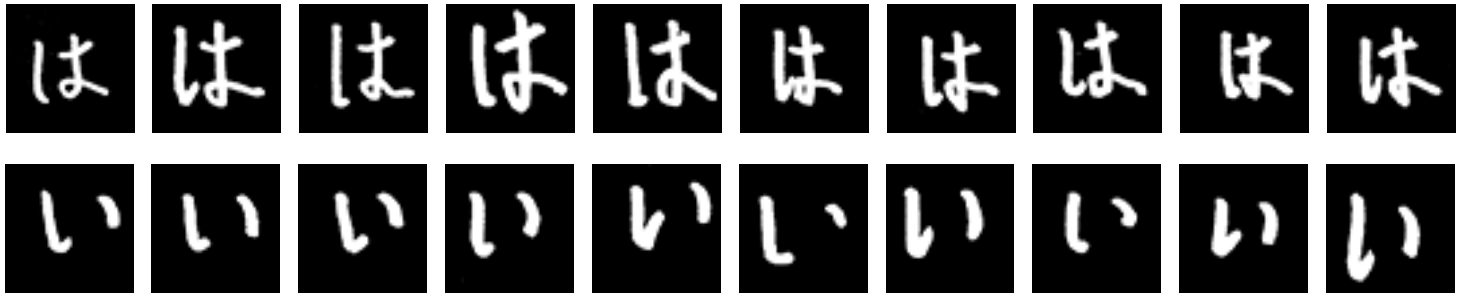
- 量子機械学習で一般的な問題を解けるか検証したい。
  （量子機械学習で実績のある二値分類で解けて、解くことに価値のある問題として筆跡鑑定を選択！）

3-2.　筆跡鑑定 -学習データ-

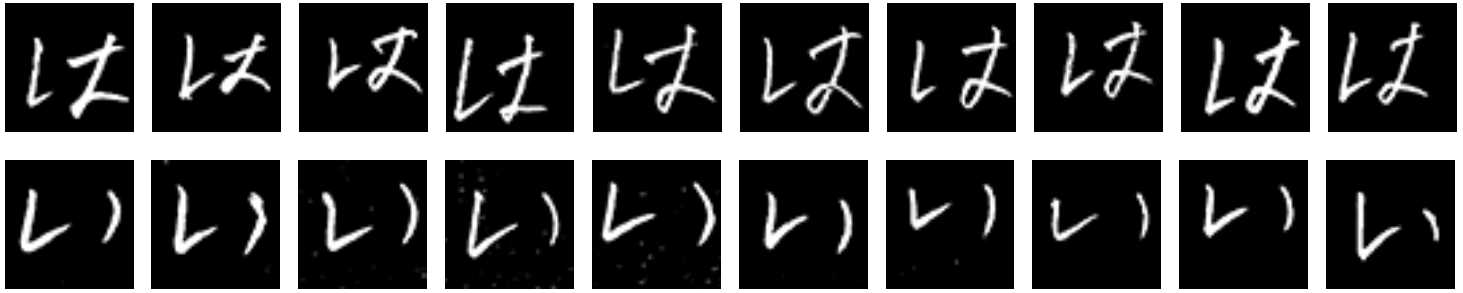
準備した学習データ

- 2名分の"は"と"い"の手書き文字をそれぞれ10個ずつ準備しオリジナル学習データとした
- ラベル：0,1
- 画像サイズ：28x28ピクセル
- 8割を学習データ、2割を評価データとした

Label:0



Label:1

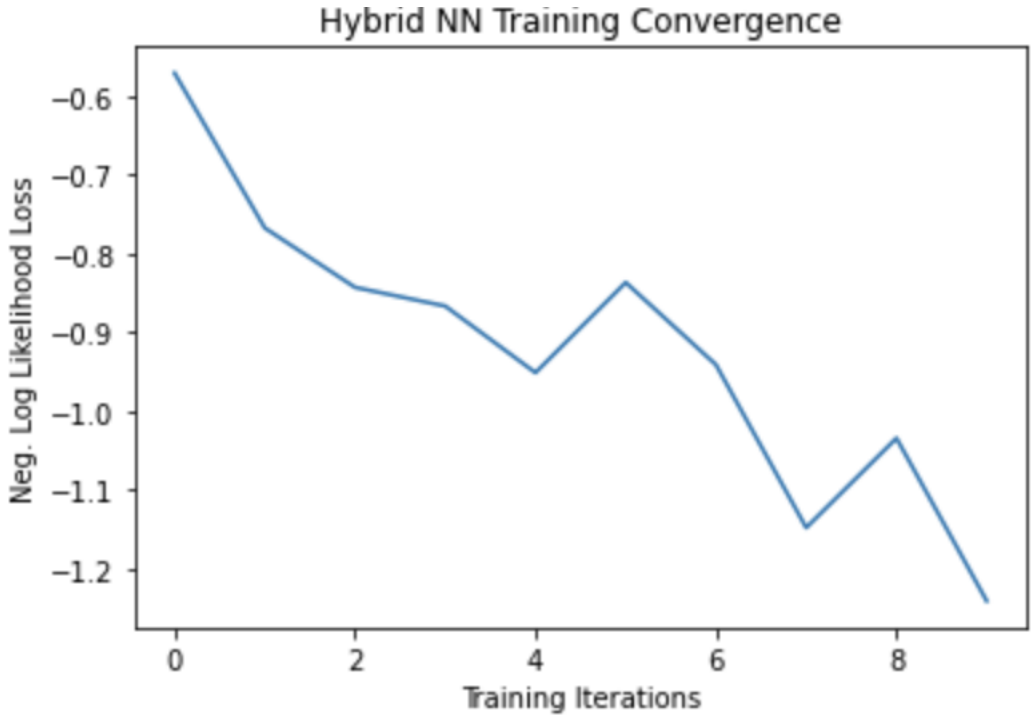


3-3. 筆跡鑑定 -学習-

学習

- 学習時の損失とグラフは以下

```
Training [10%]    Loss: -0.5703
Training [20%]    Loss: -0.7670
Training [30%]    Loss: -0.8425
Training [40%]    Loss: -0.8663
Training [50%]    Loss: -0.9509
Training [60%]    Loss: -0.8363
Training [70%]    Loss: -0.9409
Training [80%]    Loss: -1.1479
Training [90%]    Loss: -1.0341
Training [100%]   Loss: -1.2409
```

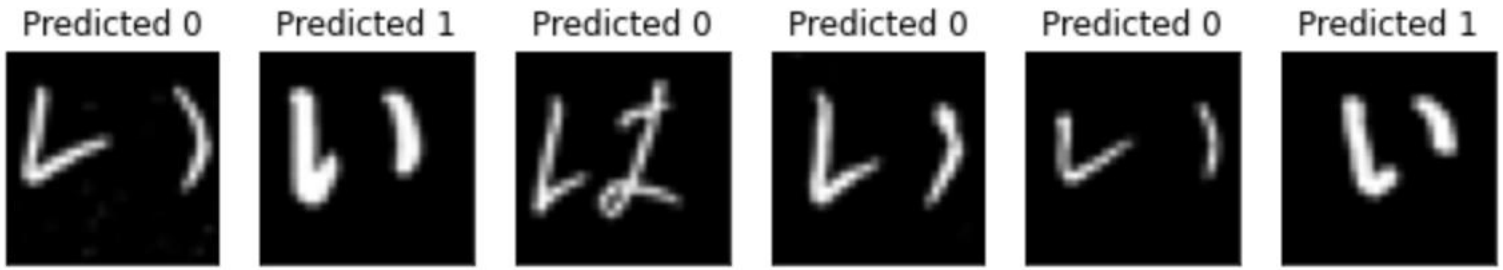


3-4.　筆跡鑑定 -評価-

評価

- スコア(損失と正確性)と検証用データの推論結果は以下。

```
Performance on test data:
Loss: -1.3129
Accuracy: 96.9%
```



人が書いた文字で作成した学習データでも量子機械学習を実施でき MINISTの学習データセットと遜色ない精度を出すことができた

量子機械学習（QNN）でも一般問題が解けた！

4. まとめ

当WGでは次の3つの成果物を作成した。

①量子コンピューター初学者向け学習資料

初学者が「Qiskit textbook」を使用して量子コンピューターを学習する際の補助となる学習教材、
および初学者が学習時に躓きやすい知識をまとめた知識集を作成した。

②量子積分アルゴリズムの実装

古典モンテカルロ積分の代替として、量子加速が得られる量子積分アルゴリズムについて、
現在単純に実装できることが知られている$\sin^2 x$系以外の関数の量子積分回路を実装した。

③日本語手書き文字認識・筆跡鑑定回路の実装

量子ニューラルネットワークを用いた日本語手書き文字のMNIST分類回路、
および2者の筆跡判定回路の実装を行った。

成果物②および③の公開場所

下記にてコードを公開
https://github.com/wg-quantum/2022-B-10a